REMARKS

Page 4 of the specification has also been amended to state, "A computer program product comprising computer program code stored on a computer readable storage medium, which when executed on a data processing system, instructs the data processing system to carry out the following method: A method for the remote and dynamic configuration of a server to facilitate capacity on demand comprising the steps of:

(a) a client device requesting a resource from a first server in a communications network;

(b) the first server receiving the request for the resource from the client device;

(c) the first server routing the client request for the resource to a dynamic content module, the dynamic content module identifying an available third server from which the requested resource can be served and routing the requested resource to the client device;

(d) collating performance data from the first and third server and the first server reporting the performance data to a second server;

(e) a second server analysing the performance data collated in step (d) to determine performance capabilities of the first and the third server and identifying if the first or the third server has reached a predetermined threshold; and

(f) the second server adjusting the allocation of the first server or the third server in response to step (e) and issuing a configuration update instruction for the first server or the third server to a dynamic configuration module of the first server and determining if a resource update is successful."

No new matter has been added, for the following reasons.

The amendment to page 4 of the specification has been copied almost verbatim from the original claim 20 and original claim 1, as originally filed.

Claims 42-49 have been canceled above, and new claims 50-62 entered.

Claims 38-41 were rejected under 35 USC 103(a) based on US 5,951,694 to Choquier et al. and US 7,080,378 to Noland et al. Applicants respectfully traverse this rejection based on the following.

Claim 38 recites a method for allocating an additional real application server to an existing pool of real application servers, said pool including a first real application server having an application installed therein and communicating with a real data source server to obtain application data from said data source server, said additional application server having said application installed therein, said method comprising the steps of:

a real management server for said pool receiving from said first real application server performance data for execution of said application in said first real application server;

said real management server, based on the performance data, automatically determining that said first real application server is functional but has reached a predetermined upper level of utilization, and in response,

said real management server automatically identifying said additional available real application server as having said application but not currently allocated to said pool, and

said real management server automatically sending connection settings for said real data source server to said additional real application server to configure said additional real application server to send subsequent requests for application data to said real data source server.

Thus, the real management server not only selects the additional real application server to add to the pool, but selects the real data source server for the additional real application server by sending connection settings for the real data source server to the additional real application server. Neither Choquier et al. nor Noland et al. taught or suggested this latter feature of claim 1. Choquier et al. disclose:

"In accordance with a dynamic load balancing feature of the invention, when a user sends a request to open a service, the Gateway microcomputer that receives the request initially identifies the application servers that are within the relevant service group. The Gateway microcomputer then determines the current load of each application server in the service group, and applies a load balancing method to select an application server that is relatively lightly loaded. The service request is then passed to the selected application server for processing." Choquier et al. Column 2 lines 48-57.

"The service map 136 advantageously serves to inform the Gateways 126 (and other components of the on-line services network 100) of the addition, deletion or change in state of any server 120 in the system. The use of the service map 136 thereby allows changes to be made to the servers 120 of the system without manual intervention. By way of example, assume that a new server 120 is added to the host data center 1-04, and is initially configured to act as a MAIL server. Shortly after being brought up, this new MAIL server will generate and send a local map 140 to the service map dispatcher 144. The service map dispatcher 144 will then broadcast the local map 140 (as part of the next service map 136) to all of the Gateways 126, allowing the Gateways to send service requests to the new MAIL server. Thus, once the new server 120 has been configured, its addition to the system is automatic, requiring no system operator intervention.

In addition to generating a service map 136, the service map dispatcher 144 maintains a central repository of information referred to as the "global registry" 145. The global registry 145 contains various information bout the present configuration of the host data center 104. For example, for each service group, the global registry 145 indicates the IDs of the servers 120 of the service group, and the identify of the Arbiter microcomputer 128 (if any) which is assigned to the service group. Of course, the global registry 145 could be maintained by some entity other than the service map dispatcher 144. Choquier et al. Column 11 line 58 to Column 12 line 18.

"The locator program 520 accesses a locally-stored cop of the service map 136 whenever an "open" request is received form a client microcomputer 102. Each time a serve 120 is selected to handle an open request, the redirectory layer 519 records the selected server's ID within a session map 522. In the course of an ongoing service sessio, the Gateway 126 accesses its session map 522 whenever a client-server message is received, and uses the session map to properly route the message." Choquier et al. Column 13 lines 29-38.

"To accommodate for such fluctuations in service usage levels, the on-line services network 100 allocates servers 120 to service groups based on service loads." Choquier et al. Column 23 lines 32-35.

"The framework for being bale to dynamically allocate servers 120 to service groups based on service group loads is set by the service map, dynamic DLL loading/unloading, hot redirection, and Arbiter features discussed above. ... enables servers 120 to be automatically transferred from service group to service group (or between a pool of unused servers and a service group); hot redirection enables a service session to be transferred from one server to another, so that a serve can be removed from a service group without prematurely terminating a service session, and the Arbiter 128 enables new servers 120 to be "rolled forward" to the same state as the other servers of a service group (for services that use dynamic, replicated data sets). Thus, with the addition of a mechanism for monitoring service group loads, servers 120 can be dynamically allocated to service groups to achieve a balanced allocation of processing resources to services. ... the process of allocating servers 120 to service groups is completely automated, with a load monitoring program (which runs, for example, on one of the administrative servers 134) deciding when and how to reallocate servers." Choquier et al. Column 23 line 49 to Column 24 line 12.

"whenever a server 120 is transferred to or from a service group, the change is reflected in the service map 136, causing the Gateways 126 to route service requests in accordance with the change. For example, when a server 120 is added to a MAIL service group, the server will inform the service map dispatcher 144 of its change in status (by transmitting its local map 140 to the service map dispatcher 144), and the service map dispatcher 144 will include the status change (along with the CPU LOAD and CPU INDEX values for the server) in the next service map 136."

However, **Choquier et al. do not disclose or even suggest that a real management server selects a real data source server for a real application server added to the pool**. Choquier et al. fail to disclose this key facet of control by a management server. Noland et al. do not fill this gap. Noland et al. disclose:

Noland et al. disclose:

"If the deployed virtual servers are close to near-term service saturation, the software will create and deploy a new virtual server with identical applications (step 504), and then direct the service request to this new server (step 505). This new virtual server can be deployed and activated via the method depicted in FIG. 3. More than one new virtual server may be deployed and activated based on the saturation condition that is detected (e.g., rate of saturation, number of servers approaching saturation simultaneously, etc.)." Noland et al. Column 5 lines 24-36.

"Referring to FIG. 3, a flowchart illustrating the process of deploying a virtual server by means of an automated software solution is depicted in accordance with the present invention. The process begins when the user logs into the software (step 301) and selects definition criteria for the newly deployed system image (step 302). The definition criteria include the following: a pool of TCP/IP addresses that the software assigns to newly deployed virtual servers, a pool of names the software assigns to the new virtual servers, and a model image that is used as a target image for the creation and deployment of the new virtual server. The model image is the current contents of memory, including the operating system and running programs. Every new virtual server is an exact copy of the model image, except for the dynamic network and server definitions that identify the new server as a unique entity. The user verifies the definition criteria and clicks a "submit" link (step 303). This link contained an imbedded common sequence that requests the software to rapidly deploy a new virtual server. Alternatively, the step of deploying the new virtual server may be automated and triggered by specified event, e.g. reaching a saturation threshold on currently deployed servers (as described in more detail below).

Furthermore, the new virtual server can actually be a subset of the original server that focuses on a critical portion of the process. This can be done in several ways. For example, the virtual server can be composed of a series of linked server processes that are individually utilized in the overall server process. When one of the sub processes becomes a bottleneck, that sub process could be cloned as necessary to eliminate the bottleneck. In response to the request from the user, the software automatically updates the VM system directory to include the new virtual server (step 304), prepares the virtual server media (disk allocations) (step 305), propagates the server model image into the new virtual server (step 306), updates the new image with local identification parameters (step 307), and then boots the new server (step 308). After the new virtual server is deployed, the end user simply clicks another link in order to interface with the new server (step 309). Alternatively, the new server may be automatically integrated into a preexisting server cluster. The entire process of deploying a new virtual server by means of the present invention can be fully automated. When done manually, it takes less than five minutes." Noland et al. to Column 3l line 56 to Column 4 line 33.

"Referring now to FIG. 4, a schematic diagram illustrating the architecture of the virtual machine environment of the automated software solution is depicted in accordance with the present invention. The present example is described within the context of the SnapVantage software solution, but it should be pointed out that the features of the present invention may be implemented by means of other software solutions. SnapVantage VM server 401 is a Virtual service machine that manages the cloning process of Linux images, i.e. Model Images 405 and 406. This cloning process uses a Shared Virtual Array Administrator (SVA) 407 in order to create array of cloned virtual servers 408. SnapVantage runs disconnected and communicates to clients 409 and 410 via TCP/IP 404. The SnapVantage web server 402 is the location of the web pages used by the SnapVantage GUI on client 409, and executes under a local Apache (or other) web server. The local deployment application 403 is the user created code imbedded in local web pages that drives specific SnapVantage functions. This component is deployed in environments that choose to allow end users to define a new virtual server. Referring

now to FIG. 5, a flowchart illustrating the process of load balancing among virtual servers is depicted in accordance with the present invention. The present invention provides a software-based solution that utilizes a communication mechanism in either direction and monitors the load status of deployed servers." Noland et al. Column 4 line 36 to Column 5 line 4.

Like Choquier et al., **Noland et al. do not disclose or even suggest that a real management server selects a real data source server for a real application server added to the pool**. Therefore, the rejection under 35 USC 103(a) based on Choquier et al. and Noland et al. should be withdrawn.

Also, Noland et al. pertain to a virtual machine environment, where virtual servers are on the same real computer are formed by cloning. So, Noland et al do not automatically add another real server to a cluster of real servers, as recited in claim 38. There would be no motivation to combine Choquier et al. with Noland et al. because of the differences between a virtual machine environment and a real machine environment. This is further reason to withdraw the rejection under 35 USC 103.

Claim 39 depends on claim 38 and further distinguishes over Choquier et al. and Noland et al. by reciting that in response to the first real application server reaching a predetermined upper level of utilization, the real management server automatically sends to the additional real application server port settings for the real data source server to communicate with the real data source server to obtain application data from the real data source server. Neither Choquier et al. nor Noland et al. teaches or suggests this feature of claim 39.

Claim 50 depends on claim 38 and further distinguishes over Choquier et al. and Noland et al. by reciting that in response to the first real application server reaching a predetermined upper level of utilization, the real management server automatically sends to the additional real application server a description of an installation path for the real data source server to support

communication with the additional real application server. Neither Choquier et al. nor Noland et al. teaches or suggests this feature of claim 50.

Claim 51 depends on claim 38 and further distinguishes over Choquier et al. and Noland et al. by reciting that the real management server is responsive in part to the performance data for the real data source server indicating capacity to support another real application server, to select the real data source server to source application data for the additional real application server and perform the step of automatically sending connection settings for the real data source server to the additional real application server. Thus, the real management server selects the real data source server to support the additional real application server **based also on performance of the real data source server, i.e. whether the real data source server has capacity to support an additional real application server**. Neither Choquier et al. nor Noland et al. teaches or suggests this feature of claim 50.

Independent claim 52 distinguishes over Choquier et al. and Noland et al for the same reasons that claim 38 distinguishes thereover.

Claims 53-56 depend on claim 52 and therefore, distinguish over Choquier et al. and Noland et al for the same reasons that claim 38 distinguishes thereover.

Claim 53 further distinguishes over Choquier et al. and Noland et al. for the same reasons that claim 39 further distinguishes over Choquier et al. and Noland et al.

Claim 55 further distinguishes over Choquier et al. and Noland et al. for the same reasons that claim 50 further distinguishes over Choquier et al. and Noland et al.

Claim 56 further distinguishes over Choquier et al. and Noland et al. for the same reasons that claim 51 further distinguishes over Choquier et al. and Noland et al.

Independent claim 57 distinguishes over Choquier et al. and Noland et al for the same reasons that claim 38 distinguishes thereover.

Claims 58-61 depend on claim 57 and therefore, distinguish over Choquier et al. and Noland et al for the same reasons that claim 57 distinguishes thereover.

Claim 59 further distinguishes over Choquier et al. and Noland et al. for the same reasons that claim 39 further distinguishes over Choquier et al. and Noland et al.

Claim 60 further distinguishes over Choquier et al. and Noland et al. for the same reasons that claim 50 further distinguishes over Choquier et al. and Noland et al.

Claim 61 further distinguishes over Choquier et al. and Noland et al. for the same reasons that claim 51 further distinguishes over Choquier et al. and Noland et al.

Independent claim 62 distinguishes over Choquier et al. and Noland et al for the same reasons that claim 38 distinguishes thereover.

Claim 62 further distinguishes over Choquier et al. and Noland et al. for the same reasons that claim 51 further distinguishes over Choquier et al. and Noland et al.

Based on the foregoing, Applicants request allowance of the present patent application as amended above.

Respectfully submitted,

Dated: 03/11/2009
Phone: 607-429-4368
Fax: 607-429-4119

/Arthur J. Samodovitz/
Arthur J. Samodovitz
Reg. #31,297